

## **REMARKS**

This application has been carefully considered in connection with the Examiner's Final Office Action dated September 4, 2008. Reconsideration and allowance are respectfully requested in view of the following.

### **Summary of Rejections**

Claims 1-37 were pending at the time of the Final Office Action.

Claims 1-10, 12, 13, 15-23, and 31-37 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 7,159,579 B2 to Sharma et al. (hereinafter "Sharma") in view of U.S. Publication No. 2002/0032783 A1 to Tuatini (hereinafter "Tuatini").

Claims 11 and 14 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharma in view of Tuatini as applied to claim 23 above, and further in view of U.S. Publication No. 2005/0223392 A1 to Cox et al. (hereinafter "Cox").

Claims 24-30 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharma in view of Tuatini as applied to claim 23 above, and further in view of U.S. Publication No. 2006/0036448 A1 to Haynie et al (hereinafter "Haynie").

### **Summary of Response**

Claims 1, 22, 32, and 37 are amended herein.

Claims 33-36 remain as previously presented.

Claims 2-12, 14, 16, 18-21 and 23-31 remain as originally submitted.

Claims 13, 15, and 17 are canceled herein.

Remarks and Arguments are provided below.

### **Summary of Claims Pending**

Claims 1-12, 14, 16, and 18-37 are currently pending following this response.

### **Response to Rejections under Section 103**

According to the United States Supreme Court in *Graham v. John Deere Co. of Kansas City*, an obviousness determination begins with a finding that **“the prior art as a whole in one form or another contains all” of the elements of the claimed invention.**

See *Graham v. John Deere Co. of Kansas City*, 383 U.S. 1, 22 (U.S. 1966).

#### **Claim 1:**

In the Final Office Action dated September 4, 2008, claim 1 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharma in view of Tuatini.

#### ***Amended Independent claim 1 recites:***

1. A method of accessing an Enterprise Java Bean (EJB) by a non-Java application within a computing environment, comprising:
  - a) **making a call**, by the non-Java application, **to a client library**, wherein the call includes input parameters, and wherein **the client library is a linkable library that is dynamically linked to the non-Java application**;
  - b) **invoking a function within the client library to construct an HTTP request** corresponding to the input parameters of the call made from the non-Java application, **wherein numeric primitive data types of the non-Java application in the calling input parameters are converted into a corresponding text representation in the HTTP request**;
  - c) passing the HTTP request from the client library to an EjbServlet;
  - d) invoking a method on an EJB by the EjbServlet based upon the HTTP request;
  - e) returning information from the invoked method from the EJB to the EjbServlet;
  - f) decoding the returned information into an HTTP response string by the EjbServlet;
  - g) **transmitting the HTTP response from the EjbServlet to the client library**; and

- h) **parsing and converting the HTTP response by the client library** into return information compatible with the non-Java application and then passing the return information from the client library to the non-Java application, **wherein text-represented numeric values extracted from the HTTP response are converted into a corresponding numeric primitive data type of the non-Java application.**

Applicants respectfully submit that the art of record does not establish a *prima facie* case of obviousness as to the pending claims because the art of record fails to teach or suggest all of the claim limitations. Specifically, Sharma in view of Tuatini fails to teach or suggest converting numeric primitive data types of a non-Java application in calling input parameters into a corresponding text representation in a HTTP request, converting text-represented numeric values extracted from a HTTP response into a corresponding numeric primitive data type of the non-Java application, parsing and converting a HTTP response, or a client library that is a linkable library dynamically linked to a non-Java application.

I. Sharma in view of Tuatini does not teach or suggest converting numeric primitive data types in calling input parameters to corresponding text representations in HTTP requests or converting text-represented numeric values in HTTP requests to numeric primitive data types corresponding to non-Java applications.

Amended claim 1 recites, “wherein numeric primitive data types of the non-Java application in the calling input parameters are converted into a corresponding text representation in the HTTP request . . . wherein text-represented numeric values extracted from the HTTP response are converted into a corresponding numeric primitive data type of the non-Java application.”

Applicants respectfully submit that no new matter has been introduced by these amendments. Support may be found throughout the specification as originally filed, including paragraph 15 and herein canceled dependent claims 15 and 17.

The Final Office Action relied on column 6, lines 1-26, and column 23, lines 16-33, of Sharma to read on the limitations of herein canceled dependent claims 15 and 17. Column 6, lines 1-26, of Sharma discloses:

Server side runtime system 114 may . . . translate requests, such as HTTP requests, to URLs . . . For example, a HTTP request generated by client 130 may be directed to a URL associated with the servlet. The request may cause the container hosting the servlet to call a method, such as a service method, of an instance associated with the servlet's class. The servlet instance may generate content based on the invocation of the service method and the container may return the content to server 110. Server 110 may, through server-side runtime system 114, serve the content to client 130 over network 120.

The server side runtime system receives a request in a Java environment from a Java client system, and translates the received request to make a server-side call to a method. In contrast, the pending disclosure teaches a client side conversion of numeric primitive data types in calling input parameters from a non-Java application to corresponding text representations in a HTTP request that occurs before the HTTP request is communicated to an EjbServlet in a Java environment.

Column 23, lines 16-33, of Sharma discloses:

Referring back to FIG. 6, once client 510 has invoked a remote method on service endpoint 555, client side runtime system 525 may process the invocation. The processing of a remote method call may include mapping the remote method call to a SOAP message representation, which may be the protocol 535 used by model environment 500 (Step 650). The mapping may include the mapping of parameters, return values, and exceptions for the remote method call to the corresponding SOAP message. Further, the mapping may include the configuration of serialization and deserialization mechanisms based on a mapping between Java types and XML data types. Next, client runtime system 525 may process the created SOAP message (Step 660). This processing may include the processing of the SOAP

message based on the mapping of the remote method call to a SOAP representation. An encoding style associated with the SOAP protocol may define how parameters, return values and types in a remote method call are represented in the SOAP message.

While Sharma may disclose mapping remote method calls to a SOAP message representation, Sharma does not teach or suggest converting numeric primitive data types of a non-Java application to corresponding text representations. The pending disclosure teaches the specific action of converting numeric primitive data types of a non-Java application to corresponding text representations in HTTP requests to enable the invoking of a method on an EJB in a Java environment. Mapping a generic remote method call from a Java environment to a SOAP message representation for a Java environment does not disclose converting numeric primitive data types in calling input parameters from a non-Java application to corresponding text representations in HTTP requests for a Java environment.

Accordingly, Sharma in view of Tuatini does not teach or suggest converting numeric primitive data types in calling input parameters to corresponding text representations in HTTP requests or converting text-represented numeric values in HTTP requests to numeric primitive data types corresponding to non-Java applications.

II. Sharma in view of Tuatini does not teach or suggest a client library, where the client library is a linkable library that is dynamically linked to the non-Java application.

Amended claim 1 recites:

making a call, by the non-Java application, to a client library . . . wherein the client library is a linkable library that is dynamically linked to the non-Java application . . . invoking a function within the client library to construct an HTTP request . . . transmitting the HTTP response . . . to the client library; and parsing and converting the HTTP response by the client library into

return information . . . and then passing the return information from the client library to the non-Java application.

Therefore, the client library performs all of: receiving a call from a non-Java application, invoking a function of the client library, constructing an HTTP request, receiving an HTTP response, parsing and converting the HTTP response into return information, and finally passing the return information to the non-Java application.

In the Response to Arguments section on page 12, the Final Office Action argues:

[T]he fact that more than one element/component maps to the claimed client library does not negate the fact that the functionality of the claimed client library are provided by the Sharma prior art. The client runtime system (Client Side Runtime System 134/ Client-Side Runtime System 525) and client side application programming interface (Client Side API(s) 135/Stub 515/Core APIs 520) of the Sharma prior art provides the functionality of the claimed client library.

Therefore, the Final Office Action is relying on the combination of elements including the client runtime system and the client side application programming interface to read on the claimed client library. However, MPEP 2173.05(a)(III) states:

In applying the prior art, the claims should be construed to encompass all definitions that are consistent with applicant's use of the term. See *Tex. Digital Sys., Inc. v Telegenix, Inc.*, 308 F.3d 1193, 1202, 64 USPQ2d 1812, 1818 (Fed. Cir. 2002). It is appropriate to compare the meaning of terms given in technical dictionaries in order to ascertain the accepted meaning of a term in the art. In re Barr, 444 F.2d 588, 170 USPQ 330 (CCPA 1971). >See also MPEP § 2111.01<

Amended claim 1 recites, "the client library is a linkable library that is dynamically linked to the non-Java application." One skilled in the art at the time of the invention would recognize that the client library may be a module of code that is linked or otherwise interfaced with the non-Java application. For example, in a UNIX environment, the client library may be a .so file. Similarly, in a windows environment, the client library may be a .dll file. As another example, the client library may also be a .h file in the C programming

language. One skilled in the art at the time of the invention will recognize that there are other types of libraries known to those of ordinary skill in the art at the time of the invention not specifically addressed in the examples above.

Furthermore, Sharma does not teach or suggest that the combination of elements including the client runtime system and the client side application programming interface is a linkable library that is dynamically linked to a non-Java application. The only disclosures of linking in Sharma are for linking a servlet class with a service endpoint class (column 3, lines 50-51; column 15, line 53; column 16, lines 11-12), linked servlet classes provided by the deployer during deployment (column 14, lines 24-25), a linked servlet class that may correspond to a configured transport binding for a specific service endpoint (column 15, lines 54-56), linking a service reference to an actual representation and configuration of a corresponding service (column 22, lines 48-50), and linking a logical service reference to an imported WSDL-based description of the service (column 22, lines 51-53). Sharma does not teach or disclose linking the combination of elements including the client runtime system and the client side application programming interface with the client.

The only disclosures of dynamic in Sharma are for stub instances that may be configured dynamically (column 8, line 21; column 8, lines 27-28; column 8, line 29), a dynamic proxy class that supports a service endpoint interface dynamically at runtime and performs serialization and deserialization of the Java and XML data types (column 8, lines 43-44, column 8, line 49-51; column 21, lines 26-27; column 23, line 11; column 27, lines 7-9), and the dynamic invocation of an operation on a target service endpoint (column 8, lines 51-52; column 8, lines 55-56; column 12, lines 18-19; column 20, lines 61-62;

column 21, line 24; column 23, lines 14-15). Sharma does not teach or disclose that the combination of elements including the client runtime system and the client side application programming interface are associated with any dynamic action, much less dynamic linking. Therefore, Sharma does not teach or suggest that the combination of elements including the client runtime system and the client side application programming interface is dynamically linked to a non-Java application.

Applicants respectfully submit that one skilled in the art at the time of the invention will recognize that Sharma's client side runtime system, the client side APIs, and the client itself are distinct elements, none of which are consistent with the ordinary meaning of the claimed client library as understood by one skilled in the art at the time of the invention. Accordingly, Sharma in view of Tuatini does not teach or suggest a client library.

III. Sharma in view of Tuatini does not teach or suggest parsing and converting the HTTP response by the client library into return information compatible with the non-Java application.

Claim 1 recites, "parsing and converting the HTTP response by the client library into return information compatible with the non-Java application."

The Final Office Action states on page 3, "Sharma is silent with reference to ... parsing and converting the HTTP response by the client library into return information compatible with the non-Java application and then passing the return information from the client library to the non-Java application."



The Final Office Action relied on paragraphs 0060, 0063, 0119, 0127, and 0138 of Tuatini to read on these limitations. Paragraph 0060 of Tuatini discloses view handlers for generically formatting and sending responses to client computers. Paragraph 0063 of Tuatini discloses an application architecture that allows a generic application program to provide its services to various client systems that use differing data formats. Paragraph 0119 of Tuatini discloses that an external client may access shared services by providing messages through a passthru component to a messaging component (see Fig. 39 of Tuatini). Tuatini discloses that the passthru component may process the messages to a format appropriate for the messaging component. The passthru component may also return responses to the requesting client in the appropriate manner. Amended claim 1 does not generically recite returning the HTTP response to a generic application in an appropriate manner, but rather specifically recites that the client library **parses and converts** the HTTP response for a non-Java application. Paragraph 0119 of Tuatini does not provide any teaching or suggestion of parsing or converting as claimed. Further, Applicants respectfully submit that the passthru component is not a client library that is a linkable library dynamically linked to a non-Java application as claimed.

Paragraph 0127 of Tuatini discloses providing XML document type definition (DTD) documents for each access interface function of a shared service. Paragraph 0127 further discloses providing a DTD for response messages that are returned to a requesting client. Applicants respectfully submit that disclosure of a DTD that describes an XML response generated by a shared service is not disclosure of parsing and converting the HTTP response as claimed. While a DTD may be used to properly

interpret tags in an XML document, a DTD itself does not parse an XML document and it does not perform conversions on data.

Paragraph 0138 of Tuatini discloses, “For each response message received, the transport connector performs the same processing discussed (e.g., translation or additional security measures) in order to send the response message back to the requesting client.” Applicants respectfully note paragraph 0136 of Tuatini discloses, “[T]he transport connector . . . then determines the form of the sub-message (e.g., JavaBean or serialized XML) received and determines if the function of the shared service can accept that form. If not, the transport connector invokes the translator to translate the sub-message as necessary.” Applicants note that the transport connector is part of a messaging component depicted in Fig. 39 of Tuatini. Accordingly this disclosed translation is occurring between the messaging component and the shared services and not at the client library as claimed. Further, Tuatini only discloses translating between, for example, a JavaBean object to an XML format or vice versa, or between an XML format to SQL calls. Tuatini does not provide any teaching or suggestion of parsing and converting an HTTP response from an EJBServlet as claimed.

In the Response to Arguments section, the Final Office Action argues that the Tuatini prior art “**identifies view handlers for formatting the responses, and invokes identified view handlers to format and send the responses to the client computers.**” (Page 14, emphasis in the original) However, disclosure of formatting a response is not disclosure of parsing and converting a response. As discussed above in section II, it is appropriate to compare the meaning of terms given in technical dictionaries in order to ascertain the accepted meaning of a term in the art. Terms in the claims must be

afforded their art-recognized accepted meaning consistent with applicant's use of the term.

Newton's Telecom Dictionary defines "parse" as:

to divide the language into components that can be analyzed . . . Parsing is divided into lexical analysis and semantic parsing. Lexical analysis divides strings into components, called tokens, based on punctuation or other keys. Semantic parsing works to define the meaning of the string once it's been broken down into individual components. (Newton, Harry, *Newton's Telecom Dictionary*. New York, N.Y., Flatiron Publishing, 23<sup>rd</sup> Edition. p. 695 TK 5102.N49 2007)

Newton's Telecom Dictionary defines "format" as "1. Arrangement of bits or characters within a group, such as a word, message, or language. 2. Shape, size and general makeup of a document." (page 412)

According to the definitions, formatting does not require performing either lexical analysis or semantic analysis. Accordingly, Sharma in view of Tuatini does not teach or suggest parsing and converting the HTTP response by the client library into return information compatible with the non-Java application.

IV. Sharma in view of Tuatini does not teach or suggest accessing an Enterprise Java Bean by a non-Java application within a computing environment.

Claim 1 recites, "accessing an Enterprise Java Bean (EJB) by a non-Java application within a computing environment."

In the Response to Arguments section on pages 14 and 15, the Final Office Action argues that:

[T]he client computers (for requesting services or making calls) of the Tuatini prior art includes both commercial software and custom-designed software, and can include both new and **legacy applications**. The client computers also include applications supporting various languages and technologies (e.g., non-Java components). Legacy applications are old

application program that continues to be used because the user does not want to replace or redesign it. The legacy applications include mainframe applications written many years ago in languages such as COBOL, PL/1, Assembler, FORTRAN, and REXX and therefore non Java applications.

Tuatini discloses legacy applications that may be shared services:

The application architecture also facilitates the development of application programs that use various services 105, such as legacy applications and database systems . . . a single executing legacy program that may communicate with many application programs). Such remote services are also referred to as "shared services" . . . shared services and clients can include both commercial software and custom-designed software, and can include both new and legacy applications . . . the transport connector (or service adapter) then sends the sub-message to the shared service through a connector interface 4115 for that type of shared service 4180 (e.g., an Oracle database application, an IBM mainframe application supporting MQSeries, an EJB component provided by a J2EE application server, an Enterprise Information System application such as on a legacy ERP system. (Paragraphs 0063, 0110, 0120, and 0137)

Tuatini also discloses that a message component invokes the legacy application:

[I]n some situations (e.g., when a message from a legacy external application is received) the message may need to be passed through a translator 4135 before being sent to the messaging component so that the message is in a format understandable by the messaging component . . . The use of a messaging component to allow various disparate clients to access functionality provided by various disparate shared services provides a variety of benefits . . . legacy applications developed before the messaging component is installed can be included as part of the system, such as placing a small wrapper or adapter around the legacy application to receive messages and to invoke the appropriate legacy application functionality. (Paragraphs 0130 and 0140)

Accordingly, Tuatini is directed to an architecture that enables invoking functionality of a legacy application, not for the legacy application to invoke the functionality of an Enterprise Java Bean. The pending disclosure teaches:

In the field of enterprise computing, much of the focus on integration has been directed at the problem of integrating non-Java systems into Java-based computing environments. ...Therefore, a need exists for a method and system to integrate the capabilities provided by EJBs with non-Java systems, especially for non-Java environments such as legacy mainframe systems that do not have CORBA support. (Paragraph 0005)

Therefore, Tuatini does not teach or suggest accessing an Enterprise Java Bean (EJB) by a non-Java application within a computing environment.

For at least the reasons established above in sections I-IV, Applicants respectfully submit that all of the limitations of independent claim 1 are not taught or suggested by Sharma in view of Tuatini and respectfully requests allowance of this claim.

**Claims Depending From Claim 1:**

Claims 2-10, 12, 13, 15-23, and 31 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharma in view of Tuatini.

Claims 11 and 14 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharma in view of Tuatini in further view of Cox.

Claims 24-30 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharma in view of Tuatini in further view of Haynie.

Dependent claims 13, 15, and 17 are canceled herein. Dependent claims 2-12, 14, 16, and 18-31 depend directly or indirectly from independent claim 1 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I-IV above, Applicants respectfully submit that all of the limitations of dependent claims 2-12, 14, 16, and 18-31 are not taught or suggested by Sharma in view of Tuatini and respectfully request allowance of this claim. Applicants respectfully submit that neither Cox nor Haynie cure the deficiencies of Sharma in view of Tuatini.

**Claim 22:**

V. Sharma in view of Tuatini does not teach or suggest wherein buffers comprise an information text string and a format text string that comprises tags that comprise a name field, a start position field, and a length field used to locate data values in an information text string.

Amended claim 22 recites, “wherein the buffers comprise a data buffer and a format buffer, wherein the data buffer comprises an information text string comprising a text representation of all information returned from the invoked method, the format buffer comprises a format text string generated by the EjbServlet comprising format information about the information returned, and the format text string comprises tags that comprise a name field, a start position field, and a length field used to locate data values in the information text string.” Applicants respectfully submit that no new matter has been introduced by this amendment. Support may be found throughout the specification as originally filed, including paragraphs 22, 47, and 53.

The Final Office Action relied on paragraph 0131 of Tuatini to read on the limitations of un-amended claim 22. Paragraph 0131 of Tuatini discloses a client component that determines the appropriate format for a sub-message, such as by converting the XML DTD for a function request to a corresponding JavaBean class. DTD files specify the parameters and the parameter types for XML messages received by a client. One skilled in the art at the time of the inventions would recognize that DTD files define how to interpret XML messages, but do not include either a start position field for a data value or a length field for the data value that may be used to locate the data value in an information text string. Therefore, Tuatini does not disclose wherein buffers comprise

an information text string and a format text string that comprises tags that comprise a name field, a start position field, and a length field used to locate data values in an information text string.

In addition to the reasons established in sections I-IV above, for at least the reasons established in section V above, Applicants respectfully submit that all of the limitations of dependent claim 22 are not taught or suggested by Sharma in view of Tuatini and respectfully request allowance of this claim. Applicants respectfully submit that neither Cox nor Haynie cure the deficiencies of Sharma in view of Tuatini.

**Claim 32:**

Claim 32 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharma in view of Tuatini.

***Amended Independent claim 32 provides:***

32. A computing system for accessing an EJB by a non-Java application comprising:

- a) **a non-Java application in communication with a client library, wherein the client library is a linkable library that is dynamically linked to the non-Java application;**
- b) **a means for calling the client library from the non-Java application wherein said means for calling the client library is used to establish communication between the non-Java application and the client library;**
- c) **an EjbServlet in communication with the client library wherein the client library comprises a function to take input parameter information from the call, embed the information into an HTTP request, and transfer the request to the EjbServlet, and wherein numeric primitive data types of the non-Java application in the input parameter information are converted into a corresponding text representation in the HTTP request;**
- d) **a means for transferring information between the client library and the EjbServlet wherein said means for transferring it is used to**

- establish communication between the EjbServlet and the client library via an HTTP protocol;
- e) the EjbServlet configured to **receive the HTTP request from the client library** and invoke a corresponding method on an EJB; and
- f) a remote method interface (RMI) for invoking methods and returning Java objects between the EjbServlet and the EJB.

Applicants respectfully submit that no new matter has been introduced by this amendment. Support may be found throughout the specification as originally filed, including paragraph 15 and dependent claim 15. Applicants respectfully submit that amended claim 32 is not taught or suggested by Sharma in view of Tuatini because they fail to teach or suggest all of the limitations recited in amended claim 32. Specifically, Sharma in view of Tuatini fails to disclose converting numeric primitive data types of a non-Java application in input parameter information into a corresponding text representation in a HTTP request, or a client library.

Amended independent claim 32 includes limitations substantially similar to the limitations discussed in sections I, II, and IV above. For at least the reasons established above in sections I, II, and IV, Applicants respectfully submit that all of the limitations of amended independent claim 32 are not taught or suggested by Sharma in view of Tuatini and respectfully request allowance of this claim.

#### **Claims Depending From Claim 32:**

Claims 33-37 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharma in view of Tuatini.

Dependent claims 33-37 depend directly or indirectly from amended independent claim 32 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I, II, and IV above, Applicants respectfully submit that all



of the limitations of dependent claims 33-37 are not taught or suggested by Sharma in view of Tuatini and respectfully request allowance of this claim.

**CONCLUSION**

Applicants respectfully submit that the present application is in condition for allowance for the reasons stated above. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, he is encourage to telephone the undersigned at (972) 731-2288.

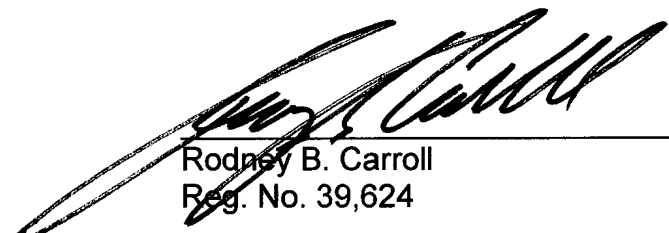
The Commissioner is hereby authorized to charge payment of any further fees associated with any of the foregoing papers submitted herewith, or to credit any overpayment thereof, to Deposit Account No. 21-0765, Sprint.

Respectfully submitted,

Date: \_\_\_\_\_

11-3-08

CONLEY ROSE, P.C.  
5601 Granite Parkway, Suite 750  
Plano, Texas 75024  
(972) 731-2288  
(972) 731-2289 (facsimile)

  
Rodney B. Carroll  
Reg. No. 39,624

ATTORNEY FOR APPLICANTS